Originally published in the April 1994 issue of Advanced Systems.

Review

Taking a dip in the software backup pool

The ASTC reviews six backup packages and finds a range of tools for every need.

By David Burnette & Cedric Higgins

Recently, one of our sister publications experienced a loss; one of its server's three disks flamed out, blowing away close to a gigabyte of data. "Just restore from backups," we told their sysadmin the next day. His blank look of shameful horror said it all. Nonetheless, he mumbled, "I was going to do them tomorrow" His voice trailed off into nothing, following the path of the 30 or 40 user accounts that were riding high on a spindle a day ago, but were now wallowing in the inner circle of data hell.

No matter how often the importance of backups is driven home to computer users, unrecoverable data loss will always occur. Over the years, disk drives have gotten more and more reliable, duping their users into thinking they can last forever. This, of course, is a fallacy brought on by technical improvements in design and manufacturing.

Aside from just plain prudence, some organizations need to archive all their work, and produce records of their vigilance regularly.

We decided to look at backup packages currently available for Hewlett-Packard HP-9000, IBM RS/6000, and SPARC systems. Since backup software is closely tied with hardware devices, we brought in a passel of tape subsystems for the software to pour data into.

The six packages we review cover a broad range of complexity and a somewhat narrower range of functionality. (We'll review two other HP, IBM, and SPARC backup packages -- Epoch Enterprise Backup and OpenVision OpenV*NetBackup -- in a future issue.) They all get the job done -- they all back up and restore files -- they just do it with an interesting array of styles and techniques.

Topping the high end of the complexity scale in this group is Spectra Logic's Alexandria. It is a backup juggernaut, comfortable at large sites with hundreds of gigabytes or more spinning. Once set in motion, Alexandria is so thorough it threatens to darn your socks and walk your dog. It's an incredible package enhanced by its seamless integration with Spectra's tape library.

Also in the big leagues are Delta Microsystems' BudTool, Legato NetWorker, and Raxco's Backup.Unet. These packages are not as intimidating as Alexandria, but suffer the pains of not being as closely mated to tape jukeboxes or carousels. There are support differences as well. Spectra Logic sells a complex product and admits installing it typically requires a call to its technical support personnel. We needed some help on several occasions and always received top-notch assistance from knowledgeable staff. The same can't be said of Legato. It implements what seems like a reasonable model for support, but in a way that was frustrating for us.

Delta Microsystems suffers not from support problems but reliability hang-ups. We crashed the package a couple of times during our tests and, after consulting with its technical support, learned a lengthy bug list is available from the company that helps avoid these pitfalls.

Nestled toward the simpler end of the scale are Workstation Solutions' Quick Restore and Software Moguls' SM-arch. These lack the install and configuration complexities, the cost, and the GUI real estate of the other packages, but there is an elegance to simplicity, and Quick Restore and SM-arch possess it.

On the hardware side, we put five kinds of tape devices through their paces (see sidebar): two tape subsystems from Andataco, a Spectra Logic tape carousel, a StorageTek DataWheel, and an Exabyte 10e tape stacker. The successful union of software and hardware is an art, and our time in the lab did not always produce masterpieces. Our advice: have technical support contacts handy. We don't mean to sound overly negative; just realize getting hardware up on its feet is sometimes no easy trick, but when everything is running smoothly, that's when you can switch from finger painting (and other gestures) to oils and brushes.

Batteries not included

We tested these packages on a pair of SPARCstation 2 clones and an IBM RS/6000. All are also available in HP-UX versions. Most require a machine with at least 16 megabytes of RAM, which isn't onerous.

The six vary widely in disk space requirements. BudTool consumes about 30 megabytes, Alexandria 30 megabytes, Backup.Unet 20 megabytes, and Networker about 12 megabytes. Quick Restore and SM-arch are the lean, mean, backup machines consuming a paltry 14 and 10 megabytes, respectively. (For a table of features, see below.)

Spectra Logic Alexandria

Burned by ravaging hordes two millennia ago, the Great Library of Alexandria has a respectable reincarnation in Spectra Logic's Alexandria. We tested Alexandria with one of Spectra Logic's 8mm tape carousel subsystems. The STL-8525/2 holds 40 tapes, contains two drives, and has a barcode reader so tapes can be tracked automatically. Luddites might set upon Alexandria with sledges, hoping to reenact a many-century's old razing. We, however, mastered and embraced its myriad capabilities.

The software for all supported servers and clients arrived on one 8mm tape, along with a three-ring binder of documentation divided into software-installation, user, reference, and hardware-installation sections. We set the tape carousel on a desktop next to an Integrix SPARCstation 2 clone that would host the software as its server. To avoid long SCSI cables and their bothersome side effects, we placed the carousel as close as possible to the server. Though not terribly loud, the unit's two fans emit enough noise to warrant it being alone in a room with the server, rather than on a user's desktop driving innocents slowly insane.

Before plunging into the installation, Alexandria's documentation requests you create an *alexbkup* user and group, as well as a home account, which serves as the location for Alexandria's executables, database, and other files. Also, the installation section begins with a detailed overview of what will transpire when you walk down the install road. Here at the ASTC, we've always touted the virtues of checklists to guide you through software minefields. Spectra Logic has gone one step further and devotes several pages to preparing both you and your system for the install. We applaud the company's efforts.

Spectra Logic prides itself on not requiring you to build a new kernel to include device drivers for the tape carousel. Alexandria controls the unit over a serial line connected to one of the server's tty ports. The software knows how to control different libraries by means of device *personalities*, which are software modules licensed by Spectra Logic that imbue Alexandria with the knowledge to manipulate a variety of robotic subsystems from Exabyte, ADIC, HP, and Odetics. However, we had to rebuild our kernel to accommodate the library's three SCSI target IDs -- one for the robotics and one for each of its two drives (the robotics require a SCSI target even though control signals travel over the serial line). With this done, we forged ahead and attached the SCSI and RS-232 cables, rebooted the server, and brought up *Xalex*, Alexandria's administration GUI. A deceptively simple menu bar greeted us.

Alexandria is a complex tool and the primary menu bar is an elegant entry into its inner workings. Six buttons line the top and activate pull-down menus to perform all manner of administrative, backup, and restore tasks: *Reference, Operation, Administration, Security, Configuration*, and *Quit*. Below the buttons are 10 status indicators, some of which turn red to warn of particular events, such as the database exceeding a certain predefined size or Alexandria's

network, database, or scheduling daemons becoming inactive.

Operation and *Administration* are the meat and potatoes of Alexandria. With *Operation*, you configure and schedule backups and restores. *Administration* manipulates tapes in the library, monitors backups and restores, and controls Alexandria's daemons.

The *Security* submenu lets you play Caesar and restrict who has access to Alexandria's many commands and operations. You can dispense privileges based on user, group, and remote-machine, or lower the portcullis completely and let no one other than the administrator enter Alexandria's hallowed halls. You add new libraries, devices, and device personalities, and set a legion of options with the *Configuration* menus. The options specify which mail program the system will use, the frequency of tape-drive cleanings, and 30 or 40 other parameters that customize Alexandria to your site.

There are two flavors of the GUI: *Xalex* for administrators of the software, and *xalex* for normal users. *Security* options must permit normal users to access Alexandria's functions for *xalex* to be of any use. Even so, it lacks the *Security* and *Configuration* buttons and sports a truncated set of *Administration* options, but does permit plebes to schedule their own backups and restores. Alexandria abides by Unix user and group permissions, so users running either of these interfaces will be able to back up and restore only those files Unix permits them to access. It is best, of course, for administrators to run *Xalex* as root and have the entire system at their beck and call.

A third GUI-based utility, *oops*, helps users recover their own files. It is essentially an interface into the restore window of the two *alexes*. Though Spectra Logic bills *oops* as an easy-to-use utility, we find it does not live up to this description. Whereas experienced Unix users and administrators might be able to safely navigate through *oops* and the steps required to conduct a restore operation, novices are likely to founder. During our testing, an editor inadvertently removed a file and wanted to recover it. He came to us and asked if we had backed up one of our servers recently. As we had backups running eight ways to Sunday, our answer could only have been yes. This was a perfect opportunity to put *oops* through its paces. The editor was able to recover his file, with a little prompting from us, but scoffed at *oops*' alleged ease of use. Nonetheless, by using *oops*, Alexandria recovered the lost file handily and without any operator intervention.

This brings us to one of the benefits of using a tape library, as opposed to standalone tape drives. Alexandria lives up to its name as a librarian. The software maintains a detailed database that tracks files it has backed up, the files each tape contains, where those tapes are located, and an abundance of other information. Throughout our testing, we dumped gigabyte after gigabyte into the tape carousel and didn't have to touch a single tape. The STL-8525 became, in effect, a 200-gigabyte bucket that gurgled on our desktop (the collection of gastrointestinal sounds emanating from the unit drew amused smirks from our office staff). Alexandria handles all of the tape-management tasks and automatically backs up its vital database, cleans the library's tape drives, and warns when tapes have been overused and should be replaced. Though pricey (the STL-8525/2 we used tops $30,000), sites that want to dispense with manually tracking tapes might find the expense worth the time and effort saved.

To recover files you must first back them up. Scheduling a backup in Alexandria requires you to complete a form the size of Kansas, called an "op-card" (short for operation card). The op-card screen is a little daunting at first, but its complexity fades with time. It's divided into thirds, the first of which sets basic parameters for the backup, such as include and exclude lists for files, users, and groups. The next two portions specify the characteristics for full and incremental versions of the backup, respectively. For these you must set the time you want the backup to run, the files you want to back up, and three important expiration dates. The expiration dates determine when the backup data will expire from the database, when the data will expire from the tape, and when the tape itself should migrate out of the library and into a storage vault. Carefully selecting these dates will establish a regular flow of data to the library subsystem and from there to long-term off-line storage. Alexandria's documentation offers some advice for properly choosing appropriate expiration times. Improper selections can waste server filesystem space and library tapes.

The full and incremental sections of the op-card also define commands that Alexandria will execute when the backup in question begins, succeeds, and (perish the thought) fails. These can be normal Unix commands, shell scripts, or other

programs.

An Alexandria backup is a process rivaling open-heart surgery. It executes backups in three phases. The first is an ordinary Unix *find* command that creates the list of files within the directory trees to be backed up. One of the op-card fields sets options for the *find* invocation, which is an easy way to skip NFS-mounted filesystems or perform whatever magic your version of *find* offers. Next, Alexandria sorts this list and prunes it according to the inclusion and exclusion rules defined in the op-card. Finally, Alexandria copies these files to the library, writing the data in *tar* or *cpio* format, or its own (based on *cpio*).

In addition to Alexandria's sprawling GUI, the package comes with an extensive set (nearly 20) of command-line utilities that can perform almost all the functions that the GUI offers. The exceptions are a few library tasks resident in the tape-manipulation submenu of *Administration*.

Like the other packages in this review, Alexandria operates over a network of heterogeneous machines. You can set up clients two ways: by NFS-mounting the executables from the server machine or by copying the necessary binaries to the clients. The easiest way to get the binaries to the client is with NFS mounts. This method also makes upgrading to new versions of Alexandria easier than if the client software were sprinkled over an entire network. An install script configures a client to work with the Alexandria server. Communication between the two is by means of Berkeley sockets. No systems files, such as */etc/services* or */etc/inetd.conf*, need to be edited to get a client running. We did amend */etc/rc.local* to start Alexandria's daemons at boot time, but this is an optional step for the sake of convenience.

Alexandria has some spiffy features above and beyond the luxury of its tape library. Files can be restored to ma- chines different from where they were backed up, and file paths can be remapped to new target directories. On the down side, its GUI is complex, requiring some effort to learn. Spectra Logic admits, in a future release, it hopes to enhance the GUI and make it more intuitive. We found Alexandria's reports and status messages to border on the abstruse and lacking an abundance of salient information. Some were so dense with machine-speak that they brought back the horrors of COBOL programming. Mitigating Alexandria's complexity is Spectra Logic's support staff. They answered all our questions quickly and deftly guided us through a couple of tricky spots in a friendly manner.

In sum, Alexandria is an ideal product for large sites that want to reduce administration overhead by using a tape library. Though complex, the GUI is a powerful means to schedule detailed backups and restores. The package is probably not the best choice for small shops with less than a gigabyte spinning. Other companies have greater marketshare and are better known, but given time, we are confident Alexandria will give the likes of NetWorker and BudTool a run for their money.

Delta Microsystems BudTool

Delta Microsystems BudTool for RS/ 6000 and SPARC arrived on two 8mm tapes, one for each platform. BudTool requires 30 megabytes of disk space, a computer with at least 16 megabytes of memory, and an ample partition for a database. The file-history database amounts to about 1 percent of the total size of all files backed up. We loaded BudTool on a Tatung COMPstation 40, which is a SPARCstation 2 clone. Our tape device was an 8mm dual-drive subsystem from Andataco.

BudTool does not require kernel modifications. Noticing that the software started daemons on our system, we decided to amend our */etc/rc.local* script to start them automatically. To our surprise, the installation script had already added them. Neither the script or the documentation said our system files would be modified. Worse yet, the script did not make a backup copy of the original *rc.local*.

Delta Microsystems offers a quick-configuration tool, called *btadmin* as well as an interface into BudTool, called *xbt*. The documentation suggests you should not launch *btadmin* until you have learned BudTool's concepts. By setting an environment variable, you can invoke *xbt* in either Motif or Open Look styles. We used Motif. Mmail

BudTool is designed with the needs of large sites in mind. It uses databases, applications, and daemons to perform backups and retrievals. The product depends on no less than seven databases: the Request database contains all of the

backup requests; the Schedule database assigns group names to a calendar; the File History database contains information regarding the files that have been backed up; the Tapes database contains media and request history; the *Configuration* database contains general BudTool configuration information; the Group Name database contains the names of all backup groups; and the Media Server database contains the media-server names, associated hostnames for each media server, and specific backup-device configurations.

Configuring the databases is simple but does require you to have the documentation handy. We appreciated the ability to set up remote and local media servers. BudTool supports various brands of jukeboxes, 8mm drives, and stackers. Laudable are the four types of reports BudTool generates.

BudTool conducts two types of backups: manual and scheduled. Our first attempt was manual, the simpler of the two. After launching the manual-backup window from our GUI, we specified a filesystem, a backup device, selected the data-storage format, and asked that BudTool generate a report upon completion. How easy is this to do? For a beginner it is not. We first needed to create a volume label (assign a name to a tape), set up a media server (a machine that has a device attached to it), set up groups (collections of backups), and determine what filesystems needed to be backed up. Once we did all this, the manual backup worked like a charm.

We then tried a scheduled backup. Our test equipment consisted of three clients and the BudTool and media server. We wanted to test the software over a bridge, so two of the three clients were part of another network, connected by a bridge to the media server. If the following procedure sounds complicated, it is. BudTool has the needs of larger sites in mind, and the operator pays a price in ease-of-learning and ease-of-use. To exercise this test, we first created two groups, one called *editorial* and the other called *astc*. We created these groups with the Group Name database edit window. Interestingly, the only item we needed to specify was the name of the group. This was the easiest of the database windows to use.

The next step was to specify which filesystems we wanted to back up. We launched the Request database edit window. Since this is the heart and soul of BudTool, we paid close attention to this database and kept the documentation handy. Its first field specifies the hostname to be backed up (we would have preferred it to say "client"). Next is the user field, which is the account name under which the backup will run, followed by the ID field, which is a simple description name that identifies the request. There are also notes and e-mail fields. Following this is the option to turn file history on or off.

Next, you select the backup's class, which determines the commands BudTool will use to process a backup request as well as the data-storage format. BudTool supports a *dump*, *tar*, and *cpio* formats across a range of vendor platforms. The documentation gives a thumbnail sketch of the situations where you should use each format. Briefly, it recommends using *dump* to back up entire filesystems or to perform incremental backups, to use *tar* to back up all the directories or files in a particular path, and to use *cpio* to back up directories or files based on specific criteria.

Next, you label and initialize a volume. You do this through the media-management window. Once you supply a name, the initialization process will create a volume label and reserve an area for the table of contents on the tape. Having initialized our 8mm media, we launched the schedule database to schedule our requests and defined our first backup.

The time we specified for our backup to start came and went and nothing happened. After rereading the documentation and calling Delta's technical support, we learned you must close the schedule-database window for a backup to start if it is scheduled in the current hour, which ours was. We reset the start time for our backup and closed the schedule-database window. At the appointed time, the backup began archiving data from multiple clients and across our network bridge.

We then turned to BudTool's restore capabilities. We think it is important that users have the ability to restore their own files with ease. BudTool succeeds here with a user-friendly tool crafted to grant users the ability to restore only those files with appropriate Unix permissions. Clever describes BudTool's ability to restore files to a remote system if the user has correct permissions. This is convenient if files need to be recovered to a machine different from which they were originally archived.

Delta Microsystems includes *btnet* with BudTool. Though space does not permit a detailed discussion, *btnet* identifies filesystems currently mounted on your network. It maintains its own database containing the names of filesystems it finds, the names of the hosts you do not want to monitor, and filesystems you do not want to back up. Making it a candidate for the Niftiness Hall of Fame, *btnet* will automatically generate backup requests for new filesystems.

We encountered several bugs in BudTool. After pointing them out to Delta's technical support, we received workarounds and were directed to a list of more than 50 known bugs in BudTool. Delta gave us an early version of a patch to BudTool that the company expects will be available to normal customers by the time you read this. We installed the patch without complaint, solving some of our problems, but not all.

Our interaction with Delta's technical support staff was at one point contentious but generally friendly and helpful. The technicians know the product well. However, we came across one technician who evidently had several bad days. We asked him several questions concerning a particular problem we were having with BudTool. His answer: "Read the manual." Not to be put off by his abruptness, we read the manual -- not once, but twice. Two days later, with much of the manual memorized, we called again. As luck would have it, we got the same terse technician. This time we were more knowledgeable about the software and our problem was solved in about 20 minutes.

Legato NetWorker

NetWorker comes in an impressive shrink-wrapped packaged emblazoned with a medieval knight and bearing little resemblance to the Unix software packaging we're used to. Instead of the customary bulky brown cardboard box, Legato went upscale with cellophane wrap and a trim photo-adorned box a little bigger than the latest Stephen King novel (hardcover).

Legato NetWorker's documentation set included an installation and maintenance guide, a user's guide, and an administrative guide. We loaded NetWorker on the aforementioned Tatung into a *networker* directory in */home*. The total install size was about 12 megabytes.

We chose a StorageTek as the target for NetWorker's backups. The carousel had a pair of 8mm drives and 54 tapes. It bears a strong resemblance to Spectra Logic's carousel device, and though it can hold more tapes, the unit's LED front-panel display is not as verbose as Spectra's LCD screen. The 9708 arrived with a full complement of tapes, and the hardest part about setting up the device was loading the darn things into the carousel. We reconfigured our COMPstation's kernel to accommodate the unit's three SCSI targets, and loaded a special Legato-provided device driver for the carousel.

Experiencing difficulties in getting NetWorker to communicate with the carousel, we contacted Legato's technical support, which gave us some tips to try. After trying theirs and a few of our own ideas, we finally got the two chatting.

After checking to make sure its daemons were running, we powered the carousel, set environment variables, moved into NetWorker's home directory and invoked *networker*. A four-section, three-buttoned GUI greeted us. Before going any further, we studied Legato's methods and buzzwords.

NetWorker is designed to back up and restore files and directories on a heterogeneous network using a client/server model, although not in as simple and elegant a fashion as Software Moguls' SM-arch. You can start backups on a scheduled or as-needed basis (equivalent to BudTool's manual backups). Recoveries are done on demand. The NetWorker server keeps an on-line index of backed-up files, which you browse to recover a file. Clients must be configured to recognize the NetWorker server. You do this by editing the local host file on the client or editing the NIS server hosts maps to add *nsrhost* as an alias for the NetWorker server. Clients can NFS-mount their executables or store them locally. With NetWorker correctly deployed around a network, users can back up and recover files and directories as they please as long as the administrator has given them proper permissions.

With all the theory behind us, we explored the many facets of NetWorker's GUI. Under the *Administration* button you set devices, groups, schedules, pools, and policies. We found a total of 12 set-up features administrators need to learn. The *Operation* button allows you to browse volumes, indexes, create tape labels, mount and unmount tapes, and

perform backups and restores. A total of nine features are available from this menu. We decided to do a manual backup as our first foray into NetWorker's client/server jungle.

Selecting *Backup* from the *Operations* menu presented us with an OpenWindows *filemgr* look and feel. We selected the files and directories we wanted to back up, then we started the backup and our files were written to tape without incident. Curiously, we were not prompted for a storage device to use, nor did we have a selection field to specify a device. For example, if there were more than one NetWorker server available, we saw no way of specifying which of the servers to use. We later learned that this information must be set through other menus found under the *Operation* button.

With a successful manual backup under our belt, we visited the core of NetWorker: the *Scheduling* submenu. Before creating our first schedule, we took a look at the five default schedules that NetWorker offers. Legato developed these based on the most common type of schedules used by its customers. We must tip our hat to Legato for giving an in-depth detail overview of each schedule in the installation and maintenance guide. Full of confidence, and with our documentation present and technical support phone number programmed into our phone, we set about scheduling our first backup.

A backup schedule specifies what level of backup NetWorker will perform for a client on each day of the week or month. NetWorker offers 11 different backup levels: a full backup, incrementals, and a level one to nine scheme. The last is synonymous with the standard Unix *dump* command, though the documentation never references *dump*. NetWorker's on-screen calendar allowed us to set up backups for each day of the month. We were able to repeat the same schedule over a weekly or monthly period. In addition, we had the opportunity to skip days in a schedule, such as weekends or holidays.

After configuring our schedule, it was time to create a group and associate that group with our newly defined schedule. Creating a group is probably the easiest feature to use in NetWorker. Selecting the *Group* option from the *Administration* menu, we received the groups window. We named the group, selected the clients and directories, set the start time for the backup, set the number of retries that the software will try to back up a client if that client cannot be backed up (because it's down, for example), and set the *autostart* variable to enable, disable or startnow. With this variable set to startnow, a group backup could be started immediately. We selected enable, which instructed NetWorker to begin the networkwide backup of the highlighted group at the scheduled time. The only drawback we saw in the groups window is that we could only set the number of retries and not set the time intervals for those retries.

Before moving on to configuring a client for backup, we heeded a tip in the manual stating we should set up our groups, schedules, policies, and directives before configuring our clients. With this in mind, we moved ahead and configured our policies. This is where things got sticky.

NetWorker maintains on-line indexes of all the files backed up for each client and an index of the files stored on each piece of media. Depending on how you set its policies, NetWorker will control automatically how long the information is retained in its on-line indexes. We selected the *Policies* option and found Legato supplies five sample policies, which are clearly documented. We created one of our own.

Once we determined our policy, we were close to having our first scheduled backup ready to go. Then, we launched our directives utility, which allowed us to select certain files (such as core) and directories (such as */dev*) we did not want backed up.

Having all other options configured the way we wanted, we were ready to set up new clients. We launched the clients window from the administration window and performed a few simple tasks. We supplied the client's name and selected the type of backup device, the schedule we wanted to use, the browse and retention policy, the directive, the groups the client should belong to, and the filesystems that we wanted to back up. We did all this in about 10 minutes. Our backup ran to completion and we danced with joy.

Our next big task was to examine the software's recovery capabilities. NetWorker allows users to recover their own

files, so we decided to give it a shot. Our test consisted of backing up our reviews editor's e-mail files and then removing them from the system with *rm -fr. (What!? -- ed.)* We then launched the NetWorker client-side software as our reviews editor. Selecting the correct files and the correct server that files should be recovered from (this information is kept in the on-line indexes), we began to recover the lost e-mail files. Thankfully, all of the files were recovered without errors, and we must admit that this test was performed after hours, without our editor knowing (until now). (*When's the next performance appraisal? -- ed.*)

During our testing of NetWorker's recovery features, we noticed some similarities with other packages, such as relocating a file or directory to a tem- porary filesystem. One slick feature is conflict-resolution. We were able to either rename recovered files or overwrite existing files when a naming conflict occurred. Our first experience trying to reach Legato's technical support staff foreshadowed deeper troubles. We needed configuration assistance, so we pulled out the manual and found Legato's support number. After three tries we found ourselves talking to a fax machine. We then dialed 411. Calling a fourth time, we reached technical support and related our earlier attempts to reach them. Unfortunately, they said, the telephone numbers for technical support and their fax machine are transposed in the documentation.

Legato's support staff is divided into two teams. The first being a front-line technical support person and the second being a back-line support person. When you call Legato's support line, you will receive a front-line person. It is this person's responsibility to answer all calls, give incident numbers, and offer assistance. If your problem is not resolved, you are dropped into a database monitored by a back-line person who will then call you back within four hours. This sounds good in theory, however, it is a judgement call on the front-line person to determine if your problem should be forwarded or not and the importance of your problem may tend to be greater in your eyes than in theirs.

Given the critical nature of these products to users' data (not to mention a system administrator's job), we quizzed a Legato official about the company's policies. It's a matter of scale, he said. Legato's software supports PCs running Novell NetWare as well as Unix iron, so the company logs more calls than its competitors. Legato's two-stage support, he said, is the best way for all questions, both trivial and complex, to be answered promptly by the technical staffer best-suited for the call.

Raxco Backup.Unet

Moving on to a peer-to-peer backup product, we installed Backup.Unet on our Tatung workstation.

The getting started guide is spaciously laid out with a spiral binding, affording an easy read. The manual begins with an overview chapter of what Backup.Unet does, then dives into a chapter on system requirements, then a brief chapter details the three steps to installing the software: validating the network, installing Backup.Unet, and configuring it. We liked the validation process, for it avoided encountering potential system problems during the software load. Part of this validation process is verifying that Backup.Unet can control the tape devices it will use to back up data. In our case, Backup.Unet could manipulate our tape jukebox, so no kernel modifications were required. The other part of the validation process verified that remote-execution permissions existed for the machines that would be running Backup.Unet.

Raxco's install script was easy to use and intelligent enough to create directories as needed.

Having to create a special group for the software was annoying. Under NIS, of course, this required only one change on the master, but in non-NIS environments, it would require editing */etc/ group* on every system on which Backup.Unet would be installed. To properly install the GUI, we also had to add an entry to the */etc/inetd.conf* file for the *maint.unet* daemon. We liked the fact that a file was provided with all the necessary information for our */etc/rc.local*, if we wanted the daemons to start at boot time. We have to mention that we had to meddle with this file, for it lacked several crucial semicolons and caused our server to hang during reboot.

Backup.Unet has X, *curses*-based, and command-line interfaces. We opted for the X GUI, and were greeted by the usual black-on-blue Motif screen with three buttons, three menus, and a message window. Already understanding backup methodologies, we wanted to exercise some of the unique features Backup.Unet has to offer. While getting

familiar with the menus and buttons in the GUI, we ran across a little problem. Whenever we selected certain menu options we received an "rpc not registered: 999" error. Having installed everything without errors, we called technical support. They knew of this problem and gave us a fix for it in minutes. We feel this information should be placed in one of the release notes.

We began our testing by creating a work list. A work list is a file that defines the areas of a system that need to be backed up. We found the easiest way to create a work list is to copy an existing one and then modify it. We then configured our backup-configuration record. This included details as to how we wanted the backup to be written (*tar, cpio*, or ANSI format), if we wanted to split files, which pool to use, which backup device to use, whether the data should be compressed or not, and other items of this nature. Once this was done we started our backup. We encountered no problems.

Backup.Unet employs a peer-to-peer architecture, eschewing reliance on a central server for processing. Because of this, we had the ability to move from node to node to recover and store data. With this design, our backups had no single point of failure on our network (except for the network itself).

Backup.Unet's *Restore Files* option recovers files. The interface is straightforward, allowing users to select files by date and time, a named dump, or dump ID. Next, you choose the host and device to be used for restoration. Then select where you want the restored files to go. Options include instructing Backup.Unet to restore directories only, reset file dates, and restore certain deleted files automatically. Finally, you must select the files you want restored.

Like the others, Backup.Unet keeps an on-line index of backed-up files. The *List* menu allows you to display a list of backed-up files either by date and time, named dump, or dump ID. The databases are kept for both local and remote machines, and can be accessed from anywhere on the network.

For the truly paranoid, Backup.Unet encrypts data automatically during network backups. You may encrypt data as it traverses the network, then store it either in encrypted or original form. Following Unix conventions, Backup.Unet limits the ability to restore files to users who have proper read or write access. Also, the administrator can control access to back up and restore functions by authorization levels.

We generated many questions for Backup.Unet's technical team. Raxco provided fast and accurate support, and in our estimation is tops in this group.

Workstation Solutions Quick Restore

On the lighter side of backup packages, there is Quick Restore from Workstation Solutions. It is the ultralight stunt plane of backup software. Whereas other packages might attempt to lift you as far away from the mud, muck, and mire of Unix, Quick Restore beckons from the trenches and reminds you how powerful command-line utilities can be. For those folks unwilling to become immersed in the shell, Quick Restore offers a shapely GUI into its restore capabilities.

Three utilities form the bulk of Quick Restore's toolbox. Two are the command-line elves, *qtar* and *qrestore*, that perform backups and restores, while the third, *qri*, is the Motif file-recovery GUI. As you might guess from its name, *qtar*'s syntax is reminiscent of that old Unix backup war-horse, tar. In fact, tar can read tapes written by *qtar*, which is a nice fail-safe in the event you have to recover files backed up by *qtar* but, alas, don't have it around.

These utilities, and a few more that get down and do the tape-device boogie, come on an 8mm tape along with two manuals, a users guide and a guide for the Motif GUI. Installing Quick Restore is a bit more complex than its simple nature would lead you to believe. After you un*tar* a few bootstrap files from the tape, an install script takes over. It speaks in clear, well-written English, and steps through the various phases of extracting the distribution from the tape, linking executables into bin directories, and building a new kernel to accommodate Workstation Solutions' high-performance tape driver. Rebuilding the kernel on our Integrix SPARCstation 2 clone went without a hitch and, in about 20 minutes, we were ready to rock and roll.

Four types of hosts compose a network within Quick Restore's purview. An administrative host stores configuration

information, a complete copy of the software, and the database files for client hosts. A media server has one or more backup devices attached to it. *qtar* sends data to it during backups and *qrestore* retrieves data from it during restores. A client host contains data to be backed up or restored. Every client host is associated with one administrative host. An operator host is where you log in and run Quick Restore. There can be more than one of each of these hosts. The documentation advises that operator hosts must be able to access all the hosts on which data to be backed up resides. Our initial installation set up one server to be all four kinds of host.

Once you set up one administrative host, you configure clients by running the install script on this host. The script copies the necessary client executables and other files to the client, remotely configures the client, and reports back with a brief status report.

We ran into only one problem with this script. It first copies an archive of client executables into */tmp* on the client and then breaks them out from there. On one of our clients, there was not enough space available in */tmp* to accept the executable archive (it's not that big, about 4 megabytes) and the script failed to bring this client into the fold. (We're not sure if this is a limitation of the script or a lesson telling us to create larger root partitions.) Workstation Solutions suggests editing the *installqr* file to change the location of the temporary directory and informed us the next version of the software is sensitive to this problem (it checks if */tmp* is too small and if it is, requests an alternate site).

For backups of remote machines to work, the client must grant rsh permissions to the calling machine. Likewise, for a client to back up to a remote drive, the machine hosting the drive must grant it rsh permissions.

A section in Quick Restore's manual provides a tutorial on how you should set up your network backups, how best to use Quick Restore from within shell scripts, and other tips to get the most from the product. We found this section to be not only educational but thoughtful and commend Workstation Solutions for considering that some of its customers may be new to the backup game and want to understand the rules a bit better.

A backup description file (BDF) defines each *qtar* backup. You can store these files anywhere, but a default directory exists within the Quick Restore hierarchy. A BDF's syntax is simple and specifies the hostname of the machine *qtar* will back up and the directories or files to include and exclude. You can use Unix regular-expression wild-carding to specify include and exclude lists. A BDF include-file directive imports a file of additional BDF commands, which comes in handy for establishing default site-wide backup strategies.

Command-line options govern *qtar*'s execution. Some of the more important specify such things as which BDF to use, which backup device to use, the backup level (a superset of Unix dump levels), whether you want *qtar* to span filesystem mount points (handy for backing up local disks or a single partition), and whether you want *qtar* to follow symbolic links.

Once you've mastered *qtar*'s command-line syntax, defining and performing backups is a fairly easy task. Those folks who may be leery of entrusting their data to a larger package -- one that sports a flashy GUI, for example -- may appreciate *qtar*'s simplicity and resemblance to *tar*.

Instructing *qtar* to back up a remote client is a simple matter of setting the remote's hostname in a BDF. Alternatively, a command-line option can specify a remote client to back up. *qtar* can back up files to a remote media server by specifying the hostname along with the device name on the command line.

We did find one nit. One of the home accounts *qtar* backed up contained a directory hard-linked to itself. Though other packages in this review adroitly side-stepped this landmine, *qtar* stepped squarely on it and, after backing up an ever-growing chain of directories (it recursed about 50 times), complained about having too many open files and moved on. We note that standard Unix *tar* also suffers this Achilles heel.

You can perform restore operations with either the *questore* command or the *qri* GUI. *questore*'s syntax is unimposing, though Unix novices may flinch a little. It does prompt for necessary information omitted from the command line, which assists neophytes in filling holes.

Either way, *questore* eventually asks which tape device you'll use and requests that you mount the necessary volumes. Though you might abhor having to use a command-line utility to restore files, *questore* is versatile, easy to use, and like many of its ilk, powerful once you learn its syntax.

questore obeys Unix file and directory permissions, so users may recover their own files as well as any other files for which they have adequate permissions. When *qtar* backs up, it stores information about the files and the machines from which they were backed up, in a database on the administrative host. *questore* can restore files to their native machines or to a new node entirely.

questore permits you to remap portions of file paths, beginning from the left. For instance, you could restore files backed up from */home/americium/isotope* to */tmp/isotope*.

The restore GUI, *qri*, is friendly, intuitive, and features on-line help. It has two modes: a basic mode, which is the simplest to use, and power mode, which offers you a finer grain of control over the restore operation. You toggle between the two modes by activating a brilliant red button enscribed with either *Go Power* or *Go Basic* depending on which mode the GUI's currently in.

qri extends Quick Restore's simplicity into the visual realm. Whereas Alexandria's screens cover enough area to wallpaper our lab, *qri* is postage-stamp petite. Complexity has its uses, and once you've mastered the Alexandria GUI, for example, you can do wonderful things, but it's also nice to walk peacefully through a rose garden rather than attend a lecture on aromaticity.

Not surprisingly, we found *qri* easier to use than some other GUIs in this review. Certainly, any experienced Unix head will have no problem restoring files with Quick Restore's utility, and the single greatest hurdle we think greenhorns will encounter is locating and mounting the correct volume into the correct drive. This done, *qri* restores files lickety-split.

Rounding Quick Restore's kit bag of tricks, Workstation Solutions provides two utilities that control and find out information about devices. A third utility is an interactive program that controls media handlers.

Though Quick Restore's install script reconfigures your kernel and links in its high-performance tape driver, Workstation Solutions also supplies a utility that will do this separately from the install process. In case either of these procedures do not work, Quick Restore's documentation contains instructions on how you can modify your kernel manually and install the driver. We appreciate the redundancy of Workstation Solutions' approach.

Quick Restore is a lean product at a lean price. If you want to wean yourself off of shell scripts and *rdump*, Quick Restore is a good first step. Version 2.0 of the product, expected to be shipping by the time you read this, promises to take a bold leap into GUI-dom -- most prominent on the company's list of enhancements is a GUI front end to all of Quick Restore's functionality.

Software Moguls SM-arch

If there is ever a blessing in conducting a review of this size, it is the occasional product that is borne aloft on a chorus of angels. SM-arch from Software Moguls is just such a package. Some other packages in this review are complex and do their job well; they amount to the M-1 tanks of the backup software army. SM-arch is a sniper by comparison. Its lightweight design belies its efficiency and effectiveness. Perhaps it should be likened more to a sniper armed with vest-pocket atomic bombs.

Stepping in line neatly with its brethren, SM-arch arrived on an 8mm tape, accompanied by a small-sized spiral-bound manual. The text is clearly written, well laid out, and pleasant to read. (Really. We're not making this up.) Whoever wrote the manual has a talent that we wish other companies would see fit to employ.

Chapter 1 is an overview of the product and discusses what SM-arch can do, the three interfaces supported (Open Look, Motif, and *curses*), and describes the layout of the GUI. SM-arch classifies its users into three categories: administrators (those that configure the software and backups), operators (essentially human tape handlers), and users

(the manual colorfully describes these folks as the "beneficiaries" -- those who are only able to extract backed-up files to which they have access).

Chapter 2 begins the install process. Step-by-step instructions, in clean unimposing sentences, guide you through the installation and testing of SM-arch on one machine (Chapter 6 covers installing SM-arch clients). The extraction process is simple and no kernel modifications are required.

The remainder of Chapter 2 steps quickly but thoroughly through a sample backup procedure that backs up */etc*. It illustrates what SM-arch can do and how (easily) the software does it.

Interspersed through this chapter are gray-shaded boxes containing information pertinent to jukeboxes under SM-arch's control. We appreciate this organizational device, for rather than clutter the pages with information concerning all types of installations, Software Moguls chose to isolate special installs into the gray regions, and let the chapter's primary flow describe the typical setup. Readers are free to ignore or obey the gray areas as their needs warrant.

After we installed SM-arch on our server, we leapt ahead to Chapter 6, eager to set up other machines for it to play with. SM-arch clients communicate with servers over TCP/IP sockets. The software backs up NFS-mounted filesystems, so if you are hesitant to take the plunge into the client/server abyss, you can NFS-mount client filesystems onto an SM-arch server and back them up there. This requires installing the software on only one system.

To establish SM-arch clients, the client software must be made available to them, via NFS mounts or by installing the executables locally. The client executables hover around 500 kilobytes for the supported Unix platforms, and rather than worrying about exporting all our NFS filesystems with correct permissions, a simpler approach was to install the executables on each client. You must also add */etc/rc.local* to the client's to start an SM-arch daemon at boot. Then, amend */etc/services* to inform the system which port number to use. The default is 956.

Once clients are configured, you can incorporate them in backups, and their drives can be brought into SM-arch's heavenly kingdom. Like the other packages in this review, SM-arch supports a heterogeneous mix of clients. We configured our RS/6000 as an SM-arch client and had a backup running on it in under 10 minutes.

SM-arch uses what it calls classes to define which files on which machines it will back up. It calls tapes *volumes*; a volume may contain one or more partitions, which are the result of a backup operation. Defining classes is a simple point-and-click fandango. You can use wild cards to include and exclude files and directories from backups. Each class is inherently able to span machines, for when you define the files to back up, you must include the name of the machine with the files and the machine that will perform the backup (the syntax is simple: *hostname:/pathname*). This is a true client/ server product.

At the heart of SM-arch is *smarch*, the GUI executable. Invoking *smarch* brings up sheer elegance. Nine menu buttons line its top, affording access to all of SM-arch's functions. The most important are *Class, Schedule, Operator*, and *Extract*. These define classes, schedule them, mount volumes, and restore files. Other buttons configure tape devices, grant various permissions to users of *smarch*, and peek at which volumes contain which files. SM-arch works with several kinds of tape jukeboxes, but be prepared to contact the technical support staffs at both Software Moguls and the jukebox manufacturer if you want to get the two communicating quickly.

Last, but by no means least, is the *Help* button. The same wonderful coherency we found in the paper documentation is evident in the on-line help, as well. We would go so far as to say that with *smarch*'s on-line help, you probably won't need the manual once you get beyond the usual assortment of installation potholes.

Restore operations (SM-arch calls them extracts) are speedy and easy to run. There are two kinds of extracts: one where you scroll through lists of stored files and another where you specify, either explicitly or with wild cards, the files you want recovered. The first is good if you don't remember a file's exact name and the second is good for restoring large numbers of files at once. SM-arch obeys Unix permission restrictions for both backups and restores. Before users can back up and recover files, however, you must define and grant them permissions from within SM-arch. A spectrum of

permissions allows you to dole out access rights to *smarch*'s general functions. For users to invoke *smarch*, they must supply a password, adding a respectable level of security to the backup and restore process.

With the GUI, SM-arch includes a slew of command-line utilities complete with man pages. Yes, man pages. Software Moguls' documentors went docu-wild and thoughtfully provided them. These programs duplicate the functionality of the GUI, but ease of use suffers.

Though lacking the Gatling-gun approach used by more elaborate packages, SM-arch is a fine tool for folks who want a backup tool that's easy to install and administer, elegantly designed, and suited for sites with moderate to large data loads.

Backing up is hard to do

Just as there is no "best" automobile for all drivers, there is no best backup software (or hardware) for all organizations. For small departments, a shell script and *dump* is fine. For slightly larger departments (or if several small groups pool money and get a friendlier, cross-platform backup utility), SM-arch and Quick Restore come at a friendly price, both in ease-of-use and financial outlay.

As a site's complexity increases, so should the capability of the backup tool. Backup.Unet is the old Marvel comic book hero Silver Surfer, riding the network server-to-server ensuring routine backups by relying on no single point of failure. Our experiences with Backup.Unet product were almost all positive.

BudTool was an early entrant in the Unix backup market, and if Usenet discussions are any indication, it's popular too. We are troubled by its many bugs, and commend Delta Microsystems for facing the music and making its defect list public knowledge. For new administrators, we recommend waiting for a less-buggy version to ship.

NetWorker is flexible and should scale well to large sites. We are less enthusiastic about Legato's support policies and learned late in the review that one of the company's larger value-added resellers became so concerned about Legato's support, the VAR took the task on itself for its customers.

Alexandria has the advantage of being closer to a turnkey solution than the others by nature of Spectra Logic selling hardware and software together as a single unit. If you don't want to play Erector Set with your backup hardware and software, Alexandria is for you. On the other hand, tying hardware and software so close together may sacrifice flexibility in the future. ■

About the authors
David Burnette and Cedric Higgins are technical editors at Advanced Systems.

Back to Story

Tape-subsystem cavalcade

No backup-software package would be complete without some device to be on the receiving end of gigabyte upon gigabyte of data. We used five devices from four vendors. The five represent three product categories. Two are tape carousels, holding upwards of 54 8mm tapes in a rotating cage, two are desktop tape subsystems with some fancy features, and the last is a tape jukebox with a capacity of 10 8mm tapes. While not a complete look at the technologies available, our list represents some popular, moderate-cost choices.

Spectra Logic STL-8525/2 (far left)

Though closely mated with its software, Spectra Logic does sell a fair amount of its tape carousels for use with backup packages other than its own. The STL-8000 series, along with its little brother the 20-tape STL-6000, covers four

flavors: one or two drives and with or without a barcode reader. The STL-8525/2, featured here, holds 40 tapes, has two 8mm drives, and includes a barcode reader. Its list price is $31,465 alone or $38,766 bundled with Alexandria backup software (one server license and up to 25 client licenses). Spectra Logic, 1700 North 55th St., Boulder, CO 80301, 303-449-7759, 303-939-8844 fax.

Andataco Encore Plus (above)

We received two different kinds of Encore Plus tape subsystems from Andataco. The Encore Plus Tape Array contained two 8mm half-height tape drives and the other, the Encore 10, replaced the bottom tape drive with a removable 1.2-gigabyte half-height disk drive. The dual-tape unit acts in several user-selectable modes. Its two drives can work in parallel, where one mirrors the data going to the other, or in tandem, where the second drive picks up when the first fills up a tape. Other modes let it directly copy one tape onto another or switch between the two drives automatically. The unit requires one SCSI ID. The removable-disk subsystem not only served as a backup device, but its disk contained the install directories of all the backup software we reviewed. The dual-tape unit lists for $8,995 and the removable-disk unit for $5,995. Andataco, 10140 Mesa Rim Dr., San Diego, CA 92121, 619-453-9191 or 800-334-9191, 619-453-9294 fax.

Exabyte 10e (far left)

An LCD menu panel puts a friendly interface on Exabyte's 10e tape jukebox. Like its brother, the 10i, the 10e can randomly load tapes into its single drive by means of a robotic mechanism. A third in the family, the Exabyte 10, accesses tapes in a sequential order. All hold up to 10 tapes, provide 50 gigabytes of storage, and an average access time of about 20 seconds or less. The 10e's alluring green menu panel permits you to test, diagnose, and configure the unit. We found the interface easy to use and convenient, especially for changing its SCSI IDs and putting the unit through a barrage of self-tests after it arrived in our lab. Exabyte sells the 10e to distributors, VARs, and OEMs. Call Exabyte for a list of retailers: 800-825-4727. Exabyte Corp., 1685 38th St., Boulder, CO 80301, 303-442-4333, 303-442-4269 fax.

StorageTek 9708 (above)

The second tape carousel to grace our lab was a 9708 from StorageTek. Chock full of 54 8mm tapes, with two drives and a barcode reader, the unit served us well, greedily drinking in a steady backup stream from several of the software products we reviewed. Its list price is $45,000. StorageTek Corp., 2270 South 88th St., Louisville, CO 80028-4310, 303-6735151, 303-673-8876 fax.

Back to Story

Back to Story

	Delta Micro	Legato	Raxco	Soft. Moguls	Spectra Logic	Work. Solns.
GUI interface	*	*	*	*	*	*
Command-line interface	*	*	*	*	*	*
Client/server design	*	*	*	*	*	*
Supports multiple Unix platforms	*	*	*	*	*	*
Encrypts data	*	*	*	*		
Compresses data in software	*	*	*	*		*
Verifies writes	*	*	*	*		*
User can restore own files	*	*	*	*	*	*
Backs up raw partition	*	*	*	*	*	
Allows wild-card file selections	*	*	*	*	*	*
One file can be split	*	*	*	*	*	*

over multiple volumes						
Backs up active partitions	*	*	*	*	*	*
Restores file to remote machines	*	*	*	*	*	*
Notification of backup failures	*	*	*	*	*	*
Centralized control & monitoring	*	*	*	*	*	*
Supports 4mm tape drives	*	*	*	*	*	*
Supports 8mm tape drives	*	*	*	*	*	*
Maintains tape inventory	*	*	*	*	*	*
Uses unique ID for each tape	*	*	*	*	*	*
Supports tape jukeboxes	*	*	*	*	*	*
Recovers data off bad tapes	*	*	*	*	*	*
Backs up multiple clients simultaneously	*	*	*	*	*	*
Generates reports	*	*	*	*	*	
Backs up Apple clients	*			*		
Backs up DOS clients	*	*		*		
On-line help	*	*	*	*		*
On-line tutorial	*	*				
Built-in backup strategies	*		*		*	
Disk space required for installation (MB)	~30	~12	~20	~10	~30	~14

Back to Story

If you have problems with this magazine, contact webmaster@sunworld.com
URL: http://www.sun.com/sunworldonline/asm-04-1994/asm-04-backup.html.
Last updated: 1 April 1994.

Speakers Sameh Elnikety

Title Cooperative Backup Systems

Location Abercrombie A227

Time September 11 @ 12:00

Abstract During my internship at SRC during summer 2000, I worked with Mark Lillibridge and Mike Burrows on building a prototype of a cooperative backup system.

In this system, each machine stores its backup data remotely among a group of other peer machines, and in return it stores equivalent amounts of data from its partners in its local file system.

This form of cooperation and distribution gives several benefits and poses several interesting challenges. Let us first consider the benefits. As the partners are independent and geographically distributed, they have independent failure modes, which is important for a backup system and corresponds to taking the backup tapes off-site. In addition, this cooperation makes the system very cost effective as the partners do not have to pay a fee to a third party for the backup service and there is no need to purchase new equipment.

As for the challenges in the design of the system, the partners do not trust each other and it is possible that some partners are down at any moment. Therefore, we had to use several techniques to ensure confidentiality, robustness, integrity, and cooperation.

As the backup data might be sensitive and is stored remotely, we used secret key cryptography to encrypt the data. In particular, we used IDEA to encrypt the backup data before sending it to the partners during a backup operation and to decrypt the data during a restore operation.

To ensure that the partners could not modify the data unnoticed, we used a cryptographic hash function to make the data blocks self-checking. So, when a machine is retrieving its data during a restore operation, it could check the cryptographic hash values in the data blocks to make sure that it is the same data that was backed up during the last backup operation.

We used erasure codes to add redundancy to the data, so it is possible to do a backup or a restore even if a few partners are down.

The possible security attacks against the system were novel and guarding against them was the hardest design issue. We used challenges to ensure that the partners of a given machine are keeping its backup data. To challenge a partner, a machine requests a randomly chosen block of data from that partner and checks it is the right block. Also, we had to impose several rules to ensure the cooperation of the partners and to prevent any partner from gaining any benefit if it does not apply the rules. For example, to prevent an attacker from exploiting one machine after the other, we imposed a commitment cost whenever a machine acquires a new partner. The machine pays the commitment cost by being forced to store its partners' data for a certain period of time without any guarantee that it can restore its own data.

We ended our study with profiling the prototype system. We found out that the costs of encryption, computing the secure hash, and applying erasure codes are much less than what we had expected.